techds-sneak-peeks /
pandas-utility-sneak-peek

Code Issues Pull requests Actions Projects Security Insights Settings

pandas-utility-sneak-peek / functional_testing_demo_notebooks / func_test_graphing_dfplot_3_PPSF_data.ipynb

techds functional_testing > new and updated Jupyter notebooks bc01803 · 3 weeks ago

```
1183 lines (1183 loc) · 463 KB
```

Preview Code Blame Raw

In [ ]:
```
'''

Let's explore the functionality of the pdutils-dfplot method using the

PPSF dataset. The dfplot method helps to streamline visualization of

data in a source dataframe. For additional details, see the Jupyter

notebook on functional testing of dfplot using a toy dataset.

'''
```

In [2]:
```
'''

Let's start with importing all the necessary modules in the first cell. Once the modules

are imported, the cell output confirms with a message that all imports have been imported!

'''

import pandas as pd
import seaborn as sns
import warnings
from pdutils.graphing.dfplot import dfplot
from pdutils.summary.dfsummarize import dfsummarize

warnings.simplefilter("ignore")

print('All imports have been imported!')
```

```
All imports have been imported!
```

In [3]:
```
'''

In this cell, we use the variable `f` to point to the PPSF AllHomes data which

has over a million records and takes a little over 50 MB of storage space.

'''

f = '../../../../../Documents/datasets/data_RE_v3_ppsqft_allhomes.csv'
```

In [4]:
```
'''

In the next cell, you will read the given data as a Pandas dataframe (df). Use

dtype=np.object_ in read_csv() if you need to designate `object` data type for columns. This

is helpful, especially, when your dataframe columns contain mixed data types.

'''

dfx = pd.read_csv(f)
print('Specified dataset read. Shape and column list is as follows:')
print(dfx.shape)
print(dfx.columns)
dfx.head()
```

```
Specified dataset read. Shape and column list is as follows:
(1255915, 6)
Index(['date', 'reg', 'zip', 'city', 'state', 'ppsf'], dtype='object')
```

Out[4]:

	date	reg	zip	city	state	ppsf
0	2010-02-01	612	612	Thoreau	NM	NaN
1	2010-03-01	612	612	Thoreau	NM	NaN
2	2010-04-01	612	612	Thoreau	NM	NaN
3	2010-05-01	612	612	Thoreau	NM	NaN
4	2010-06-01	612	612	Thoreau	NM	NaN

In [5]:
```
'''
```

```
It appears that `reg` and `zip` represent the same geographical location. You can use the

pdutils-dforder method to select and order specific columns in the dataframe.

'''

from pdutils.tidying.dforder import dforder
dfz = dfx.dforder(columns=['date', 'city', 'state', 'zip', 'ppsf'])
print(dfz.shape)
print(dfz.columns)
dfz.head()
```

```
(1255915, 5)
Index(['date', 'city', 'state', 'zip', 'ppsf'], dtype='object')
```

Out[5]:

	date	city	state	zip	ppsf
0	2010-02-01	Thoreau	NM	612	NaN
1	2010-03-01	Thoreau	NM	612	NaN
2	2010-04-01	Thoreau	NM	612	NaN
3	2010-05-01	Thoreau	NM	612	NaN
4	2010-06-01	Thoreau	NM	612	NaN

In [6]:

```
'''

Let's summarize the data to see unique value counts.

'''


summary = dfz.dfsummarize(date_col_ls=[], num_col_ls=[],
                          to_sci_no=False, sort_mtable=False)
mtab, _ = summary
mtab
```

```
    Dataframe shape (r, c): (1255915, 5)
    Features (cols) list  : ['date', 'city', 'state', 'zip', 'ppsf']
    Missing value count   : 216973

*** Searching data for missing values...
*** Eliminating missing values...
*** Row count has dropped from 1255915 to 1038942 due to null removal.
*** Post-row-removal null count across columns is 0.
*** Getting ready to compute the stats array...
*** Using ['zip', 'ppsf'] as the list of numerical columns for the stats array.
*** Parsing numerical data ['zip', 'ppsf'] for potential outlier values (POVs)...
******************************************************
>>> Time elapsed : 0.0 sec
>>> Drop values list generated...
>>> Time elapsed : 0.6034 sec
>>> Drop values list generated...
>>> Time elapsed : 1.246 sec
>>> Drop values list generated...
>>> Time elapsed : 1.7661 sec
>>> Drop values list generated...
>>> Time elapsed : 2.1925 sec
>>> Drop values list generated...
>>> Time elapsed : 2.5555 sec
>>> Drop values list generated...
>>> Time elapsed : 2.847 sec
>>> Drop values list generated...
>>> Time elapsed : 3.0876 sec
>>> Drop values list generated...
>>> Time elapsed : 3.2707 sec
>>> Drop values list generated...
>>> Time elapsed : 3.4153 sec
>>> Drop values list generated...
>>> Time elapsed : 3.5365 sec
>>> Drop values list generated...
>>> Time elapsed : 3.6395 sec
>>> Drop values list generated...
>>> Time elapsed : 3.7294 sec
>>> Drop values list generated...
>>> Time elapsed : 3.8109 sec
>>> Drop values list generated...
>>> Time elapsed : 3.8816 sec
>>> Drop values list generated...
>>> Time elapsed : 3.9407 sec
>>> Drop values list generated...
>>> Time elapsed : 3.9906 sec
>>> Drop values list generated...
>>> Time elapsed : 4.0327 sec
>>> Drop values list generated...
>>> Time elapsed : 4.0692 sec
>>> Drop values list generated...
>>> Time elapsed : 4.101 sec
>>> Drop values list generated...
>>> Time elapsed : 4.1285 sec
>>> Drop values list generated...
>>> Time elapsed : 4.1511 sec
>>> Drop values list generated...
>>> Time elapsed : 4.1703 sec
>>> Drop values list generated...
>>> Time elapsed : 4.1875 sec
>>> Drop values list generated...
>>> Time elapsed : 4.202 sec
>>> Drop values list generated...
>>> Time elapsed : 4.215 sec
>>> Drop values list generated...
>>> Time elapsed : 4.2267 sec
>>> Drop values list generated...
>>> Time elapsed : 4.237 sec
>>> Drop values list generated...
>>> Time elapsed : 4.2459 sec
>>> Drop values list generated...
>>> Time elapsed : 4.2537 sec
>>> Drop values list generated...
>>> Time elapsed : 4.26 sec
>>> Drop values list generated...
>>> Time elapsed : 4.266 sec
>>> Drop values list generated...
>>> Time elapsed : 4.2722 sec
>>> Drop values list generated...
>>> Time elapsed : 4.2785 sec
>>> Drop values list generated...
>>> Time elapsed : 4.2844 sec
>>> Drop values list generated...
>>> Time elapsed : 4.2903 sec
>>> Drop values list generated...
>>> Time elapsed : 4.296 sec
>>> Drop values list generated...
>>> Time elapsed : 4.3017 sec
>>> Drop values list generated...
>>> Time elapsed : 4.3072 sec
>>> Drop values list generated...
>>> Time elapsed : 4.3133 sec
>>> Drop values list generated...
>>> Time elapsed : 4.3188 sec
>>> Drop values list generated...
>>> Time elapsed : 4.3242 sec
>>> Drop values list generated...
>>> Time elapsed : 4.3292 sec
>>> Drop values list generated...
>>> Time elapsed : 4.3336 sec
>>> Drop values list generated...
>>> Time elapsed : 4.3375 sec
>>> Drop values list generated...
>>> Time elapsed : 4.3407 sec
>>> Drop values list generated...
>>> Time elapsed : 4.3436 sec
>>> Drop values list generated...
>>> Time elapsed : 4.3463 sec
>>> Drop values list generated...
>>> Time elapsed : 4.3489 sec
>>> Drop values list generated...
>>> Time elapsed : 4.3512 sec
>>> Drop values list generated...
>>> Time elapsed : 4.3533 sec
>>> Drop values list generated...
>>> Time elapsed : 4.3552 sec
>>> Drop values list generated...
>>> Time elapsed : 4.3569 sec
>>> Drop values list generated...
>>> Time elapsed : 4.3584 sec
>>> Drop values list generated...
>>> Time elapsed : 4.3598 sec
>>> Time elapsed : 4.3598 sec
>>> Drop values list generated...
>>> Time elapsed : 4.361 sec
>>> Drop values list generated...
>>> Time elapsed : 4.3621 sec
>>> Drop values list generated...
>>> Time elapsed : 4.3631 sec
>>> Drop values list generated...
>>> Time elapsed : 4.3638 sec
>>> Drop values list generated...
>>> Time elapsed : 4.3645 sec
>>> Drop values list generated...
>>> Time elapsed : 4.365 sec
>>> Drop values list generated...
>>> Time elapsed : 4.3656 sec
>>> Drop values list generated...
>>> Time elapsed : 4.366 sec
Breaking as no index was found at iter 63 ...
******************************************************
>>> Time elapsed : 0.0 sec
>>> Drop values list generated...
>>> Time elapsed : 0.5204 sec
>>> Drop values list generated...
>>> Time elapsed : 1.058 sec
>>> Drop values list generated...
>>> Time elapsed : 1.5333 sec
>>> Drop values list generated...
>>> Time elapsed : 1.9588 sec
>>> Drop values list generated...
>>> Time elapsed : 2.3307 sec
>>> Drop values list generated...
>>> Time elapsed : 2.6382 sec
>>> Drop values list generated...
>>> Time elapsed : 2.9111 sec
>>> Drop values list generated...
>>> Time elapsed : 3.1416 sec
>>> Drop values list generated...
>>> Time elapsed : 3.3399 sec
>>> Drop values list generated...
>>> Time elapsed : 3.5119 sec
>>> Drop values list generated...
>>> Time elapsed : 3.6594 sec
>>> Drop values list generated...
>>> Time elapsed : 3.7875 sec
>>> Drop values list generated...
>>> Time elapsed : 3.8986 sec
>>> Drop values list generated...
>>> Time elapsed : 3.9965 sec
>>> Drop values list generated...
>>> Time elapsed : 4.0802 sec
>>> Drop values list generated...
>>> Time elapsed : 4.1514 sec
>>> Drop values list generated...
>>> Time elapsed : 4.2152 sec
>>> Drop values list generated...
>>> Time elapsed : 4.2705 sec
>>> Drop values list generated...
>>> Time elapsed : 4.3188 sec
>>> Drop values list generated...
>>> Time elapsed : 4.3597 sec
>>> Drop values list generated...
>>> Time elapsed : 4.3945 sec
>>> Drop values list generated...
>>> Time elapsed : 4.4252 sec
>>> Drop values list generated...
>>> Time elapsed : 4.4533 sec
>>> Drop values list generated...
>>> Time elapsed : 4.4773 sec
>>> Drop values list generated...
>>> Time elapsed : 4.4972 sec
>>> Drop values list generated...
>>> Time elapsed : 4.5146 sec
>>> Drop values list generated...
>>> Time elapsed : 4.5296 sec
>>> Drop values list generated...
>>> Time elapsed : 4.5428 sec
>>> Drop values list generated...
>>> Time elapsed : 4.5539 sec
>>> Drop values list generated...
>>> Time elapsed : 4.5636 sec
>>> Drop values list generated...
>>> Time elapsed : 4.5719 sec
>>> Drop values list generated...
>>> Time elapsed : 4.5792 sec
>>> Drop values list generated...
>>> Time elapsed : 4.5855 sec
>>> Drop values list generated...
>>> Time elapsed : 4.591 sec
>>> Drop values list generated...
>>> Time elapsed : 4.5959 sec
>>> Drop values list generated...
>>> Time elapsed : 4.6001 sec
>>> Drop values list generated...
>>> Time elapsed : 4.6038 sec
>>> Drop values list generated...
>>> Time elapsed : 4.6069 sec
>>> Drop values list generated...
>>> Time elapsed : 4.6096 sec
>>> Drop values list generated...
>>> Time elapsed : 4.612 sec
>>> Drop values list generated...
>>> Time elapsed : 4.6142 sec
>>> Drop values list generated...
>>> Time elapsed : 4.6161 sec
>>> Drop values list generated...
>>> Time elapsed : 4.6177 sec
>>> Drop values list generated...
>>> Time elapsed : 4.6191 sec
>>> Drop values list generated...
>>> Time elapsed : 4.6203 sec
>>> Drop values list generated...
>>> Time elapsed : 4.6214 sec
>>> Drop values list generated...
>>> Time elapsed : 4.6223 sec
>>> Drop values list generated...
>>> Time elapsed : 4.6232 sec
>>> Drop values list generated...
>>> Time elapsed : 4.6239 sec
>>> Drop values list generated...
>>> Time elapsed : 4.6246 sec
>>> Drop values list generated...
>>> Time elapsed : 4.6252 sec
>>> Drop values list generated...
>>> Time elapsed : 4.6256 sec
Breaking as no index was found at iter 53 ...
*** There are no missing elements in the target list...returning an empty index list.
*** Done POVs!
*** Attempting to compute variance inflation factors (VIFs)...

Legend   :
   C-Type: Column data type
   V-Type: Column value data type
   MLS   : Maximum length of [column values converted into] string data type
   NIL   : Null index list
   SCIL  : Search column index list (list of col-specific row indexes that match the search criterion)
   Mid   : Median
   POV   : Potential outlier value(s) for z-score thresh=1.5
   VIF   : Variance inflation factor(s)
```

Out[6]:

	Feature	C-Type	V-Type	MLS	#NonNull	#Unique	#Null	SCIL	Min	Mid	Max	POV	VIF
0	date	object	[str]	10	1255915	115	0	N/A	NaN	NaN	NaN	NaN	NaN
1	city	object	[str]	25	1255915	5344	0	N/A	NaN	NaN	NaN	NaN	NaN
2	state	object	[str]	2	1255915	51	0	N/A	NaN	NaN	NaN	NaN	NaN
3	zip	int64	[int64]	5	1255915	10921	0	N/A	612.00	44095.00	99901.00	[612, 623, 692, 693, 725, 745, 778, 791, 802, ...	1.882
4	ppsf	float64	[float64]	7	1038942	64309	216973	N/A	10.46	126.92	2799.42	[10.46, 10.58, 11.08, 11.59, 11.73, 11.98, 11....	1.882

In [6]:

```
'''
```

```
Let's subset the dataframe, limiting data poins to 3 cities: New York, Los Angeles, and

Houston.

'''

tri_city_ppsf = dfz[dfz['city'].isin(['New York', 'Los Angeles', 'Houston' ]) ]
print(tri_city_ppsf.shape)
tri_city_ppsf.head()
```

```
(33580, 5)
```

Out[6]:

	date	city	state	zip	ppsf
133975	2010-02-01	New York	NY	10001	NaN
133976	2010-03-01	New York	NY	10001	NaN
133977	2010-04-01	New York	NY	10001	NaN
133978	2010-05-01	New York	NY	10001	NaN
133979	2010-06-01	New York	NY	10001	NaN

In [7]:

```
'''

Let's summarize the tri_city_ppsf data.

'''



summary = tri_city_ppsf.dfsummarize(date_col_ls=['date'], stats=False,
                                    to_sci_no=False, sort_mtable=False)
mtab, _ = summary
mtab
```

```
    Dataframe shape (r, c): (33580, 5)
    Features (cols) list  : ['date', 'city', 'state', 'zip', 'ppsf']
    Missing value count   : 8469

*** Searching data for missing values...
*** Eliminating missing values...
*** Row count has dropped from 33580 to 25111 due to null removal.
*** Post-row-removal null count across columns is 0.
*** Getting ready to check for datetime pattern in ['date'].
>>> Checking for datetime pattern in 'date' using %Y-%m-%d as the date format...
    Done in 0.58 sec.
*** Column that comprises data with a likely datetime pattern: ['date'].
*** Done converting column 'date' to datetime dtype.
*** Parsing and converting numeric columns...
*** Columns that comprise data with a likely numerical pattern: ['city', 'state', 'zip', 'ppsf'].
An exception of type ValueError occurred (Unable to parse string "New York" at position 0)...Skipping conve
rsion of column 'city' to numeric dtype...
An exception of type ValueError occurred (Unable to parse string "NY" at position 0)...Skipping conversion
of column 'state' to numeric dtype...
*** Done converting column 'zip' to numeric dtype.
*** Done converting column 'ppsf' to numeric dtype.

Legend   :
   C-Type: Column data type
   V-Type: Column value data type
   MLS   : Maximum length of [column values converted into] string data type
   NIL   : Null index list
   SCIL  : Search column index list (list of col-specific row indexes that match the search criterion)
```

Out[7]:

	Feature	C-Type	V-Type	MLS	#NonNull	#Unique	#Null	SCIL
0	date	object	[str]	10	33580	115	0	N/A
1	city	object	[str]	11	33580	3	0	N/A
2	state	object	[str]	2	33580	4	0	N/A
3	zip	int64	[int64]	5	33580	292	0	N/A
4	ppsf	float64	[float64]	7	25111	19798	8469	N/A

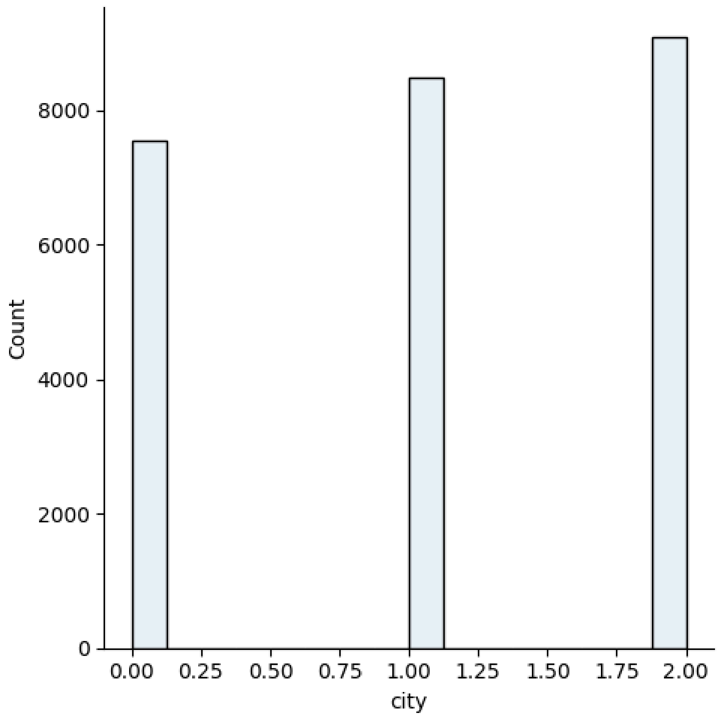
In [7]:

```
'''
```

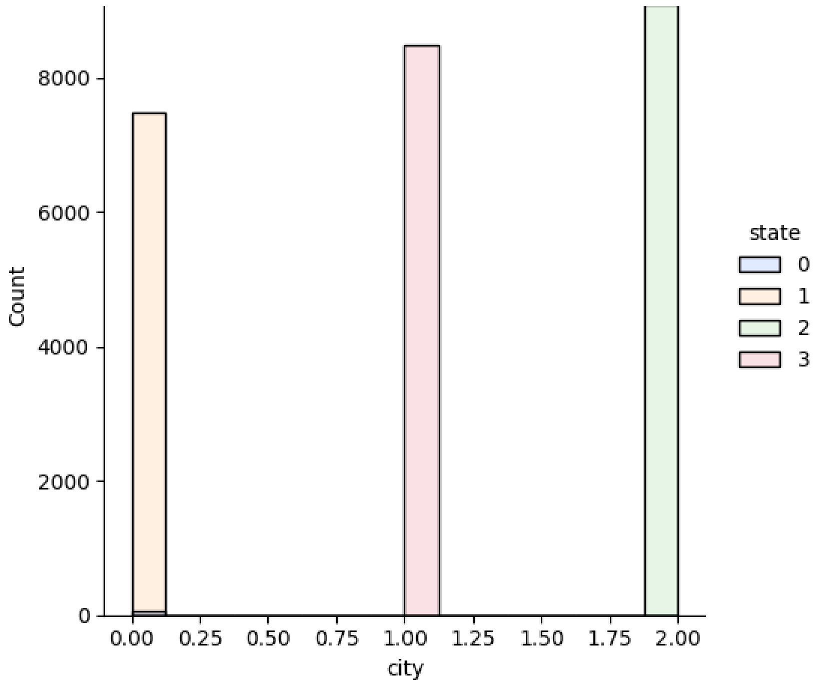
```
Now visualize tri_city_ppsf by specifying `city` and `state` as categorical columns,

i.e., cat_cols=['city', 'state']. Set show_enc=True if you want to see categorical

encodings. Below, the KDE plot on the diagonal shows that the density curve for Houston

has the highest peak. This indicates a greater concentration of properties within a

specific price per square foot (PPSF) range compared to New York and Los Angeles.

Include save_plot=True in dfplot() to save all plots.
'''

tri_city_ppsf.dfplot( cat_cols=['city', 'state'], show_enc=True)
```

```
*** Warning! Dropping rows with null values in the dataframe and resetting index...
*** Done! Row count has dropped from 33580 to 25111.
*** Checking for datetime pattern in object column data...
*** Found no object dtype columns in ['city', 'state'] that comprise data with a likely datetime pattern.
An exception of type ValueError occurred (Unable to parse string "New York" at position 0)...Skipping conve
rsion of column 'city' to numeric dtype...
An exception of type ValueError occurred (Unable to parse string "NY" at position 0)...Skipping conversion
of column 'state' to numeric dtype...
encoding: True
enc_type: cat
*** Warning! Perform encoding only after null values in the dataframe have been treated.
Generating a mask comprising a list of 35 strings.
Generating a mask comprising a list of 35 strings.
*** Warning! Since no columns to encode are specified, encoding is proceeding autonomously...
*** Proceeding with category encoding of col 'city'...
*** Done!
*** Proceeding with category encoding of col 'state'...
*** Done!
*** Mapping of column values to categorical encodings:
          city  city_id state  state_id
0  Los Angeles      2.0    CA         2
1     New York      1.0    NY         3
2      Houston      0.0    TX         1
3          NaN      NaN    MO         0
*** Dataframe column 'city' assigned to cat in dfplot_pair() within dfplot().
*** Dataframe column 'state' assigned to cat in dfplot_pair() within dfplot().
An exception of type IndexError occurred (cannot do a non-empty take from an empty axes.)...
```

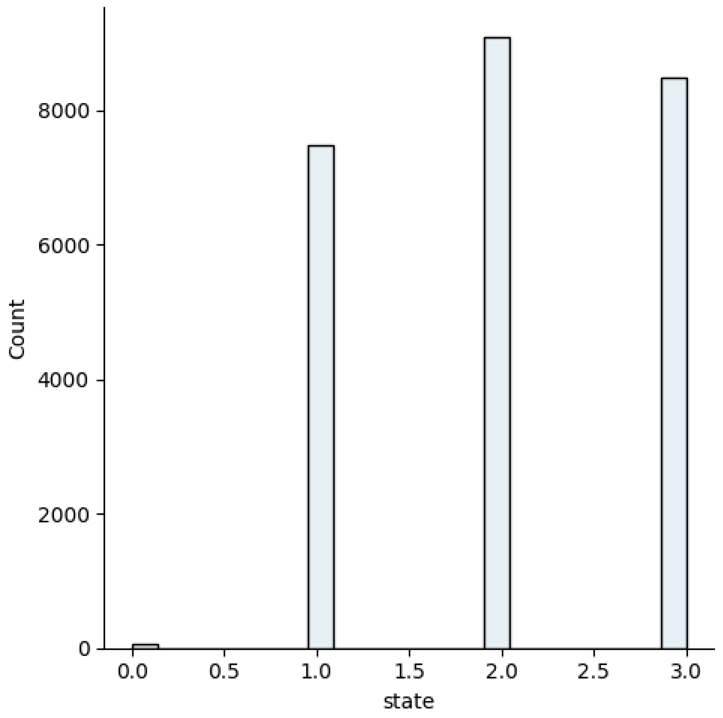
Count plot



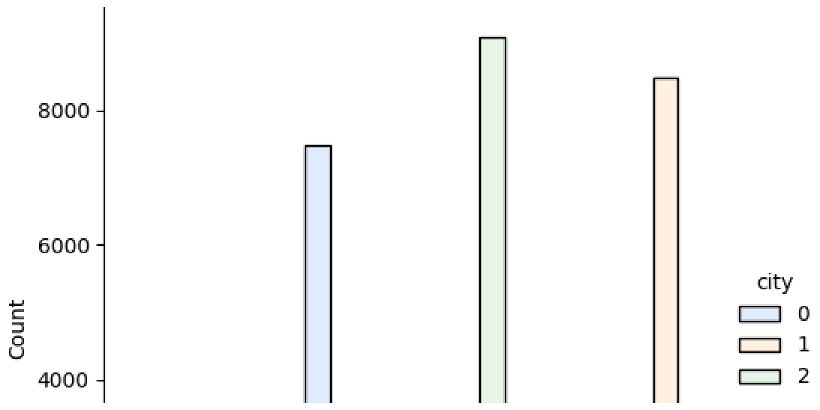
Count plot

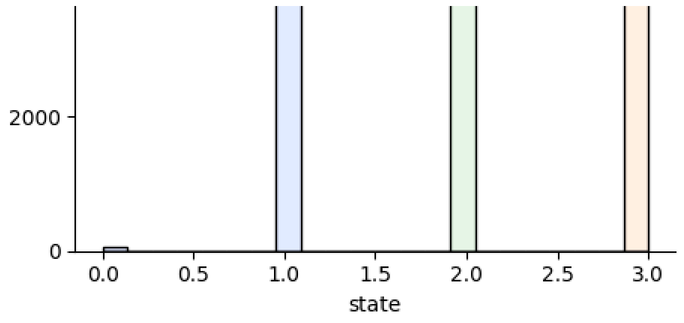


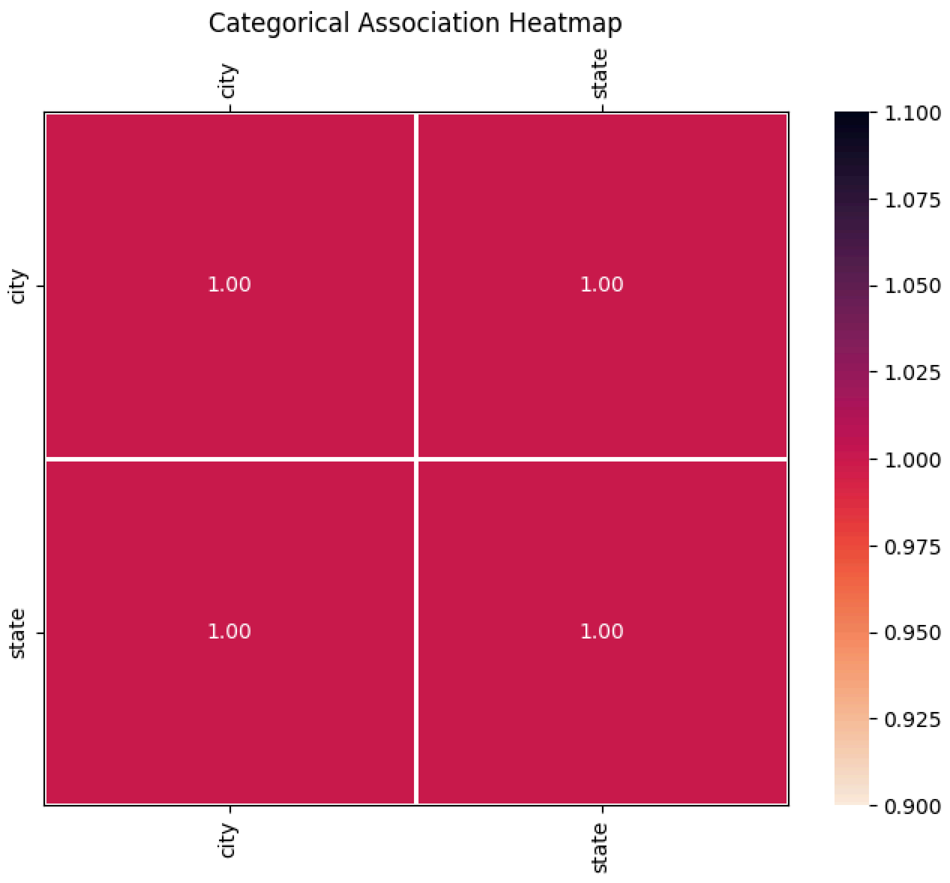
Count plot

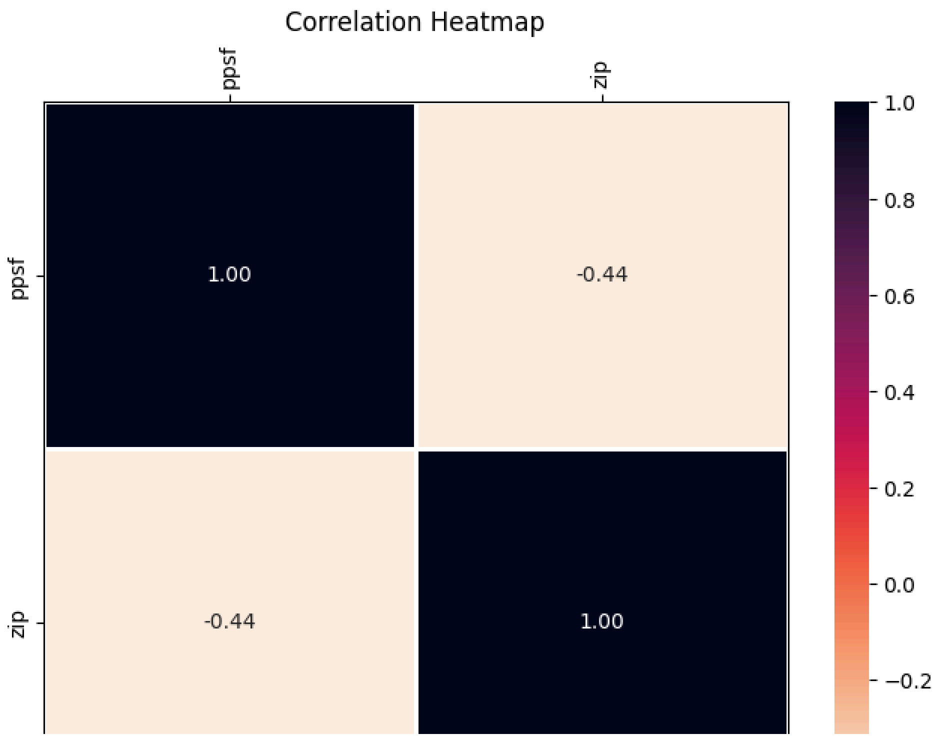


Count plot

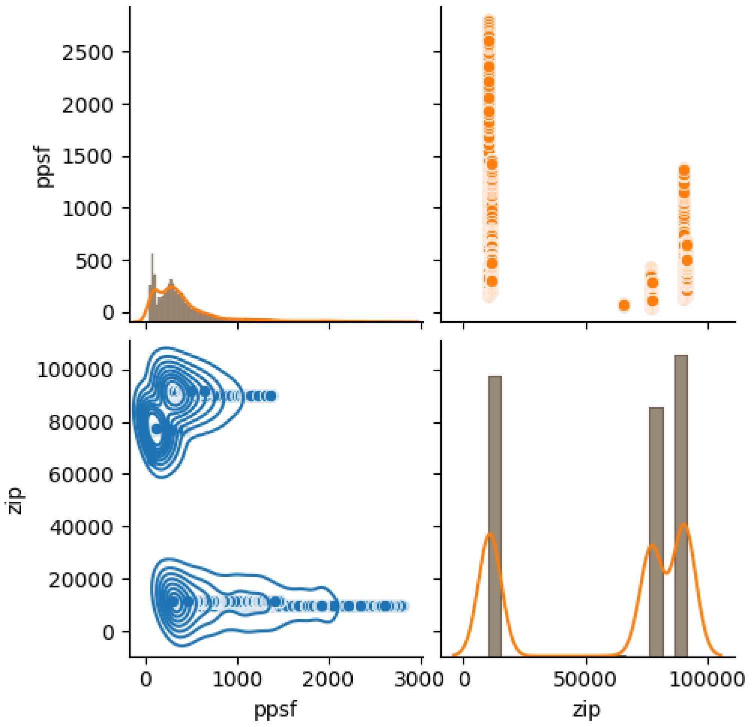



2000
0
0.0
0.5
1.0
1.5
2.0
2.5
3.0
state

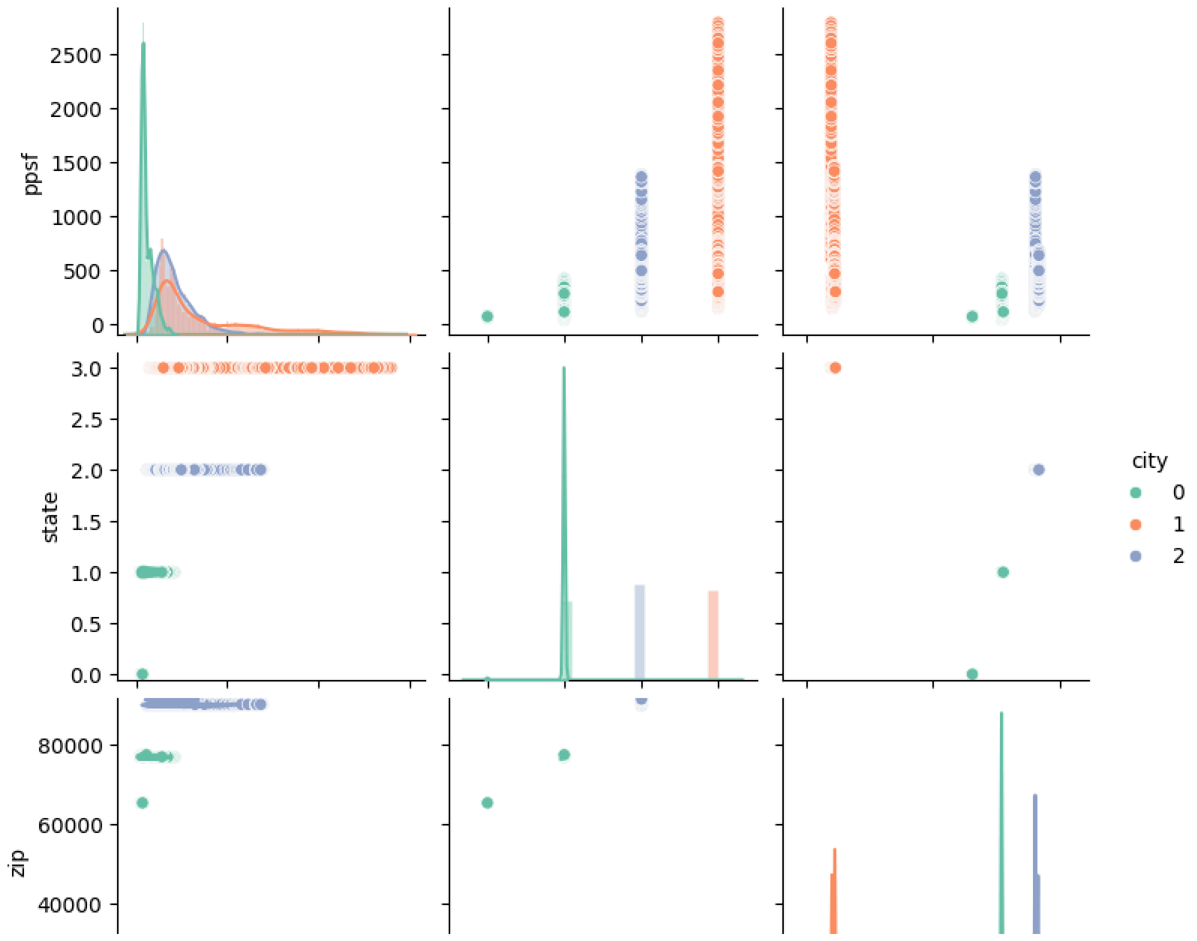

Categorical Association Heatmap
city
state
1.00
1.00
1.00
1.00
1.100
1.075
1.050
1.025
1.000
0.975
0.950
0.925
0.900

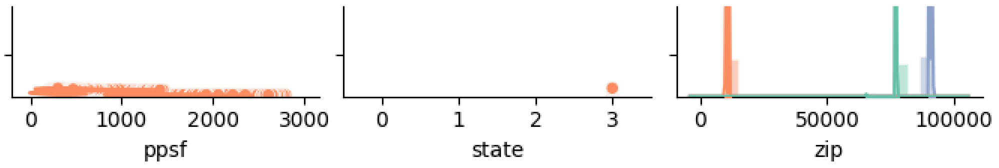

Correlation Heatmap
ppsf
zip
1.00
-0.44
-0.44
1.00
1.0
0.8
0.6
0.4
0.2
0.0
−0.2

Pair plots: Scatter (upr tri) | Histogram w/KDE (diag) | 2D KDE (lwr tri)

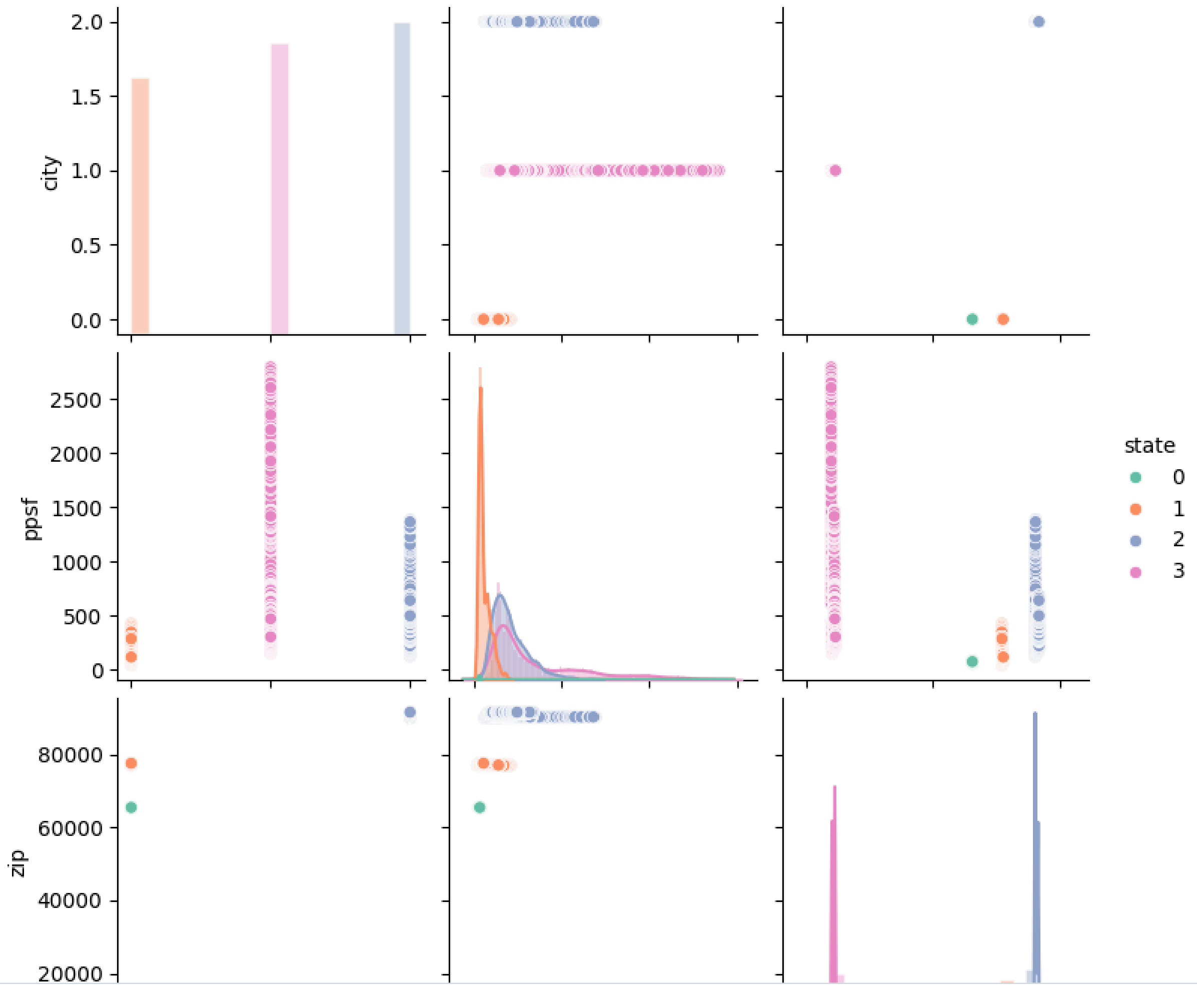


Pair plots: Scatter (upr tri) | Histogram w/KDE (diag) | 2D KDE (lwr tri)





Pair plots: Scatter (upr tri) | Histogram w/KDE (diag) | 2D KDE (lwr tri)


city
ppsf
zip
state